LBL-24862

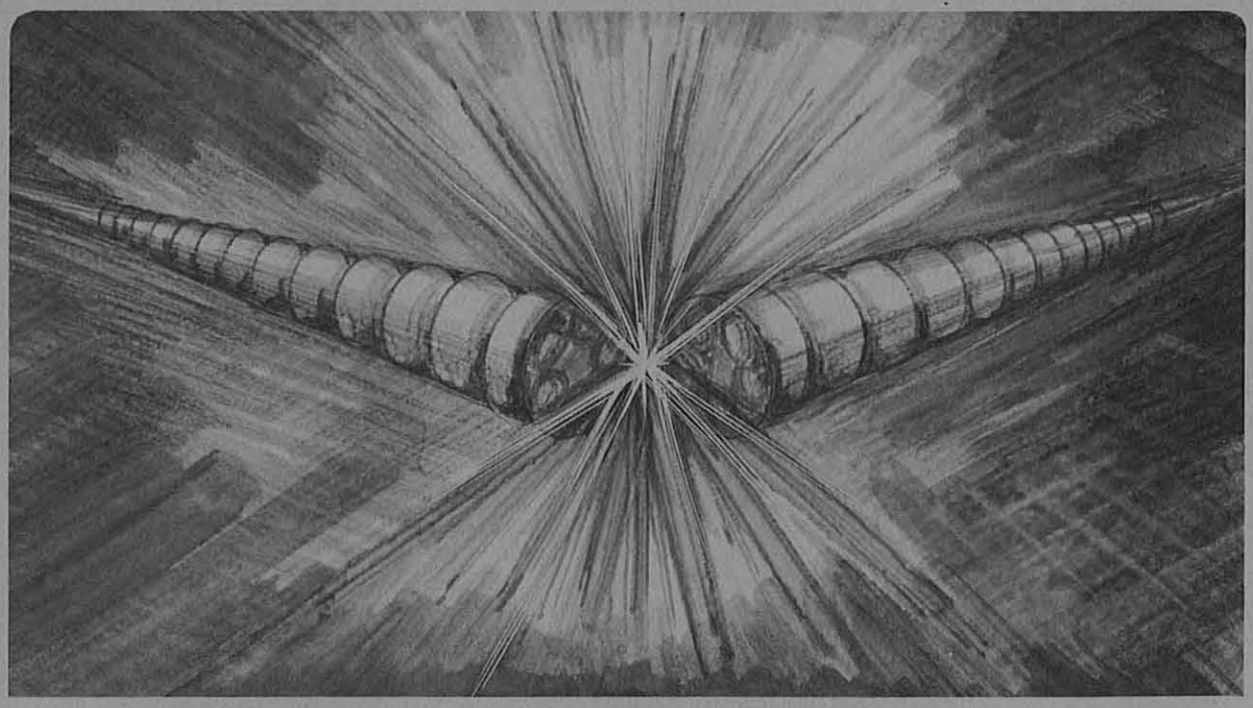
Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Accelerator & Fusion
Research Division

Interfacing AutoCAD with Magnetic Design

M. Sorin and S. Caspi

February 1988



Prepared for the U.S. Department of Energy under Contract DE-AC03-76SF00098

DISCLAIMER

SSC-MAG-187
LBL-24862

INTERFACING AUTOCAD WITH MAGNETIC DESIGN*

Mendel Sorin

ADA-Rafael
Haifa, Israel

Shlomo Caspi

Lawrence Berkeley Laboratory
University of California
Berkeley, CA 94720

February 19, 1988

*This work was supported by the Director, Office of Energy Research, Office of High Energy and Nuclear Physics, High Energy Physics Division, U.S. Dept. of Energy, under Contract No. DE-AC03-76SF00098.

SSC-MAG-187
LBL-24862

INTERFACING AUTOCAD WITH MAGNETIC DESIGN*

Mendel Sorin and Shlomo Caspi

February 19, 1988

ABSTRACT

This report is a summary of work done towards developing an AutoCAD based system for design and analysis of magnets.

The computer programs that have been developed are an attempt to integrate the new SUN computer based system with existing software on the old HP1000 System. We believe this is a good start for the further development of the whole system.

The programming languages used are AutoLISP for the programs used by AutoCAD, and Fortran (Microsoft Fortran) for all others.

The entire work has been done on IBM-AT, with the well known limits of the memory, speed of execution and operating system, therefore, some adjustment may be needed for the more powerful SUN system.

1. Introduction

The idea behind the development of the programs presented in this report, is to use a CAD system for design and analysis of magnets. The whole system is expected to provide tools for interactive design and analysis, at different levels of development. Furthermore, visualizing the magnet geometry and the possibility of changing it interactively, substantially reduces the time needed to arrive at different configurations, therefore, the process of decision making (at the research group level) will improve considerably.

So far, by using these programs, we are able to draw a magnet cross section for a "cos θ" configuration [such as NC9 for the Superconducting Supercollider (SSC)], including an iron boundary. Based on such an outlay, we can interactively calculate

the magnetic field at any desired point (except inside the conductors) and evaluate the multipoles. By using AutoCAD commands, the shape of the magnet cross section (as well as the iron radius) can be changed very easily, and the magnetic field and/or the harmonics are calculated for the new shape. The files which have been compiled are listed below.

- magnet.mnu
- poly.lsp
- field.lsp
- inspt.lsp
- dxout.for
- mgfld.for
- harm.for
- init1.dwg
- init2.dwg
- inlay.dwg
- outly.dwg
- rec4-jin.dwg
- rec4-jout.dwg
- box4-jin.dwg
- box4-out.dwg

Additional files, as listed below, are supplied by the existing system or are created during the processing.

- insrt.ang
- cor1.res
- cor2.res
- point.cad
- data.fil
- turns.cor
- poi.dxf
- poidx.res

2. Working Procedure

After loading a new drawing (working in acad.dir) the first step is to load the specially designed menu "magnet":

Command: menu Prompt: magnet

(There is no need for loading this menu for an old drawing.) Loading the magnet menu can be done automatically by AutoCAD by adding it into AutoCAD menus.

The main magnet menu consists of six submenus, namely:

- magnet
- cond-typ
- assembly
- fields
- save
- bye

2.1 magnet

This submenu sets up the environment for the entire drawing, and loads the special files to be used.

2.2 cond-typ

This submenu will contain a library of different conductor types. It is possible that the best thing to use will be the features of the "Icon Menu" of AutoCAD. A drawing will appear on the screen for each type of conductor listed in the library. So far, this submenu is a regular screen menu that contains the three submenus listed below.

- rec4-j - The trapezoidal-type for constant current density
- rec4-i - Empty
- box-typ - The conductor is represented by a little box in the trapezoid center.

2.3 assembly

This is the drawing menu, and contains four submenus.

- cos-teta - Draws the turns, the inner and outer layer boundaries, and the iron shell for "cos θ" type, magnet based on output from the program PK.
- window - An empty submenu - for further developments.
- others - As above.
- last - Returns the previous menu to the screen.

2.4 Fields

This is the submenu used for magnetic field and harmonic computations, and it contains four submenus.

- save-geo - This submenu "freezes" the actual drawing and computes the turn's coordinates and iron radius.
- calc-b - Computes the magnetic field, at any desired point (which may be entered either numerically or by means of the pointing cursor.
- harmonics - Computes the transfer function and the harmonics for the actual shape.

- last - Returns the previous menu to the screen.
- save - Saves the drawing, without quitting.
- bye - Quits AutoCAD, after asking the user to either save or discard the drawing.

3. The Basic Blocks For Drawing

The drawing procedure (which is explained later) uses two drawing files in order to draw the magnet cross section: inlay.dwg and outly.dwg. These drawings contain a geometry of a typical turn of the inner layer and the outer layer, respectively. After such drawings have been made, a dxfile is written from which data is extracted for the cross section construction. Inlay.dwg and outly.dwg must always have the same drawing form. Following is an explanation of how this is accomplished:

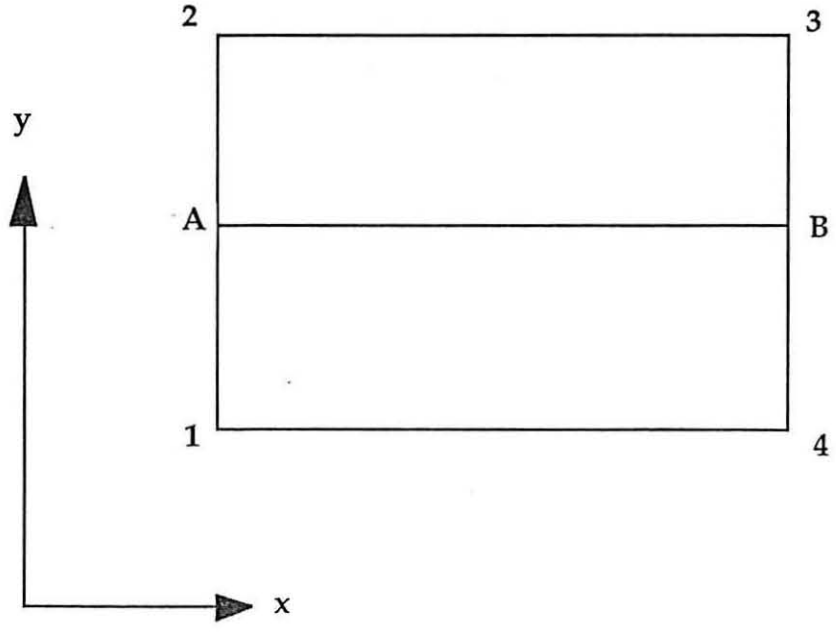
Inner Layer

Step 1. - Draw the outline of a single conductor on the inner layer. (4 vertices going CW), use x y coordinates for any turn from the program PK.

Step 2. - Make a block by the name init1 which contains the drawing in Step 1, and choose the insertion point as the inner layer O.D. (point B).

Step 3. - Make a drawing file init1.dwg through the command wblock.

Step 4. - Make a new drawing using the command "Insert init1." The rotation angle for insertion will be the angle which brings the line A-B parallel to the X-Axis.



Step 5. - Make a block which contains the drawing from Step 4, and name it by a proper name (which will represent the conductor type on the inner layer).

Step 6. - Like Step 3, with the name from Step 5.

Outer Layer

- Repeat Steps 1 through 6 for the outer layer conductor, but instead of init1, the name will be init2, and the name of the final block will be representative for the outer layer.

During the work with the magnet Menu, choosing a conductor type, the program will automatically load the appropriate drawing files and load them into inlay.dwg and outly.dwg.

4. The Files

4.1 Magnet Menu

This file contains special commands which have been designed for drawing and computations. As explained earlier, one must load this file in order that the menu will appear on the screen. In the future, this file should be added to the AutoCAD Menu in order to make it one of the Main Menu submenus. The main features of this menu are listed below.

setvar "cmdecho" 0 - Inputs and prompts are not echoed.

vmon - Is used to save heap and should be removed when used with Unix operating system.

setvar "mennecho" 1 - Menu input is not echoed.

[rec4-j] - Once this submenu is chosen, the old working files inlay.dwg and outly.dwg are overwritten by the working files for the current choice of conductor.

[box-type] - As above.

[save-geo] - A dxfile, by the name of poi.dxf is nested (using dxfout command), and a procedure "dxout" is activated (through the shell command).

[calc-b] - The special design getz command allows the user to choose any desired point for field computation.

After the operation has been completed, the x,y coordinates of the chosen point are written on the "point.cad" file, and the procedure "mag.fld" is activated (through the shell command).

[harmonic] - The procedure "harm" is activated (through the shell command).

[new-Rad] - The radius can be changed through the following steps. First, the entire drawing is brought to the screen by the "zoom extent" command. This is required so that the present iron boundary will be shown on the screen. The second step is to activate the procedure "ironchg", which allows the computer to find out the present iron radius.

The next step is to change the iron radius, either numerically or by the cursor. That is done using the change command, when pointing on the existing iron boundary as the entity to be changed. After the change has been made, a dxfile is created (through the dxfout command) and the procedure dxout is activated (through the shell command).

4.2 poly.lsp

This file contains the procedures (in AutoLISP language) used to draw the turns. The procedures are listed below.

4.2.1 drawp

This procedure is defined through drawp, allowing the AutoCAD to recognize drawp as a usual command of the system. The procedure opens the file "insrt.ang" for reading the insertion and the rotation angle, respectively, and then draws the turns by inserting the blocks inlay.dwg and outly.dwg. The insertion point for each turn is defined, in polar terms, by the radius of the inner layer or the outer layer and the insertion angle. The procedure always checks whether the rotation angle in use is greater than the previous one. If so, the insertion point is computed using the inner layer radius (pradin) and the inlay block. If the above condition is not true, the outer layer radius (pradout) and the outly block are used instead.

4.2.2 arcbound

The procedure arcbound is used for drawing either the inner and outer layer boundaries, or the iron shell boundary. The input for this procedure is the radius of the desired layer.

4.2.3 The Additional Files

The file contains five more procedures, namely, fbl, fbl1, eatblank, car4str, and cdr4str, which have been written by Michael Helm.

fbl finds the next blank in the string, using the recursive procedure fbl1. eatblank cancels the blanks in a string, until a nonblank character is found. car4str returns the first nonblank characters of a string. cdr4str returns the last nonblank characters of a string. car4str and cdr4str are similar to the AutoLISP functions car and cdr, which apply to elements of list (instead of strings).

4.3 field.lsp

This file contains three procedures which are involved in the process of drawing and computing, and are written in terms of AutoLISP.

4.3.1 getz

By activating this procedure, the user is asked to choose the desired point at which the magnetic field should be computed. After the point is chosen (either by entering x,y coordinates or by pointing), the coordinates are written on file "point.cad", which is the output of this procedure. "point.cad" is the input for the procedure "mgfld.for" explained later.

4.3.2 getr

The procedure getr allows the user to choose the desired iron radius and, accordingly, the iron boundary is drawn (by means of the arcbound procedure). In addition, an output file "iron.rad" is created, containing the iron radius.

4.3.3 ironchg

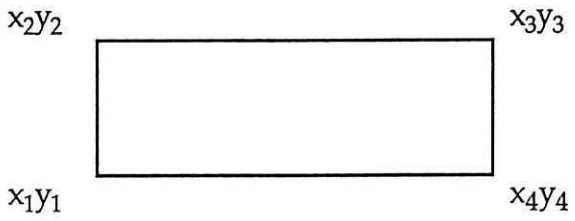
This procedure opens the file "iron.rad", reads the actual iron radius, and allows the computer to locate the entity to be changed (namely, the iron boundary).

4.4 inspt.lsp

This file contains the procedure anglist and the auxiliary procedures car4str, cdr4str, fbl, fbl1, and eatblank (which have been explained in section 4.2.3).

4.4.1 anglist

This procedure computes the polar angle ptang of the insertion points, and the rotation angle rotang of each inserted turn. While the polar angle is in radians (as required by the polar point definition in AutoLISP), the rotation angle is in degrees, as required by the insert command of AutoCAD. The input for the computations is the file turns.cor, which contains the x,y coordinates of the turns' corners.



$$\text{ptang} = \tan^{-1} \frac{(y_3 + y_4)}{(x_3 + x_4)} .$$

$$\text{retang} = \tan^{-1} \frac{(y_3 + y_4) - (y_2 + y_1)}{(x_3 + x_4) - (x_2 + x_1)} .$$

The angles are written (as strings, through the AutoLISP function angtos) on the file "insrt.ang", which is the input for "poly.lsp". The language used is AutoLISP.

4.5 dxout.for

This file contains the procedure "polygon." After the drawing has been completed, and a dxfile has been created, this procedure is used in order to find out the new coordinates of the turns' corners (if any change has been made) and the iron radius.

The program searches the file poi.dxf (the output of dxfout command) until the first insert line. Then, the parameters of the insertion points (coordinates and angle) of the basic blocks are written on a temporary file "ins.res" by means of the subroutine par finder. The process is terminated when the string entiti is reached. Then, the writing is done on the temporary file cor1.res for the corners (coordinates and rotation angle) for the inner layer, until the string outly is reached. Then, the writing is continued on the temporary file "cor2.res for the outer layer until the string circle is reached. When that happens, the next data is the iron radius which is transferred to the file "iron.rad." As mentioned above, the searching and writing is done by the subroutine par finder.

After the process of searching was completed, the temporary files ins, cor1, and cor2 are used as input for computing the new coordinates of the corners. This is done by the subroutines dxcalc and corner calc, and the functions delta x calc, delta y calc. dxcalc computes the dx and dy for the corners of each basic block, and transfers them to a coordinate system rotated by the rotation angle.

corner calc computes the corners' coordinates of each inserted turn by adding the adjusted dx, dy (for each corner) to the insertion point of the turn.

delta x calc and delta y calc compute the dx and dy, respectively, which are rotated by the rotation angle. The new coordinates are written (each turn on a line) on file "poidx.res," which is the input for mgfld and harm.

4.6 mgfld.for

This file contains the procedure "magnet field" which calculates the magnetic field B*, B*image and (B* + B*image), at any desired point Z. The files listed below are the input files for this procedure:

poidx.res	-	Contains the corner coordinates.
point.cad	-	Contains the Z coordinates.
data.fil	-	Contains the permeability end current values for a specific design.
iron.rad	-	Contains the iron radius.

4.7 harm.for

This file contains the procedure for calculating the harmonics and the transfer junction for a certain design. The original procedure was developed by L. J. Laslett, and the present one is a slightly modified version, adapted to the input procedure the entire system is using. In addition, the procedure calculates and prints the transfer function.

The input files for this procedure are poidx.res, data.fil, and iron.rad, which have been presented in the preceding section.

4.8 Drawing Files

The drawing files, which are listed below, are used either for drawing the magnet cut (through the poly.lsp procedure) or for constructing the basic elements of the drawing (namely, the conductor shape).

init1.dwg - A temporary file, used to store the basic element for each new conductor-type constructed (on the inner layer).

init2.dwg - As above, for the outer layer.

inlay.dwg - the file is used to store the inner layer conductor, when the drawing and the analysis is performed.

outly.dwg - As above, for the outer layer.

rec4-jin.dwg - Represents the inner layer conductor of the NC-9.

rec4-jout.dwg - Represents the outer layer conductor of the NC-9.

box4.jin.dwg - Represents a small box around the middle of the inner layer conductor of the NC-9. This type is used when analyzing the behavior of the magnetic field as the current is concentrated in the middle of the conductor, instead of being spread over the entire area.

box4-jout.dwg - As above, for the outer layer conductor.

4.9 By-Product Files

The files listed below are created by some of the procedures mentioned earlier and are used as input by the others. In spite of the fact that these files have been explained in connection with the creation files, we will review them briefly.

insrt.ang - Contains the insertion and rotation angle for the turns.

ins.res - A temporary file, storing the insertion parameters of the elementary blocks.

cor1.res - A temporary file, storing the corner coordinates and the insertion angle of the inner layer turns.

cor2.res - As above, for the outer layer turns.

point.cad - Stores the coordinates of the desired point at which the magnetic field should be computed.

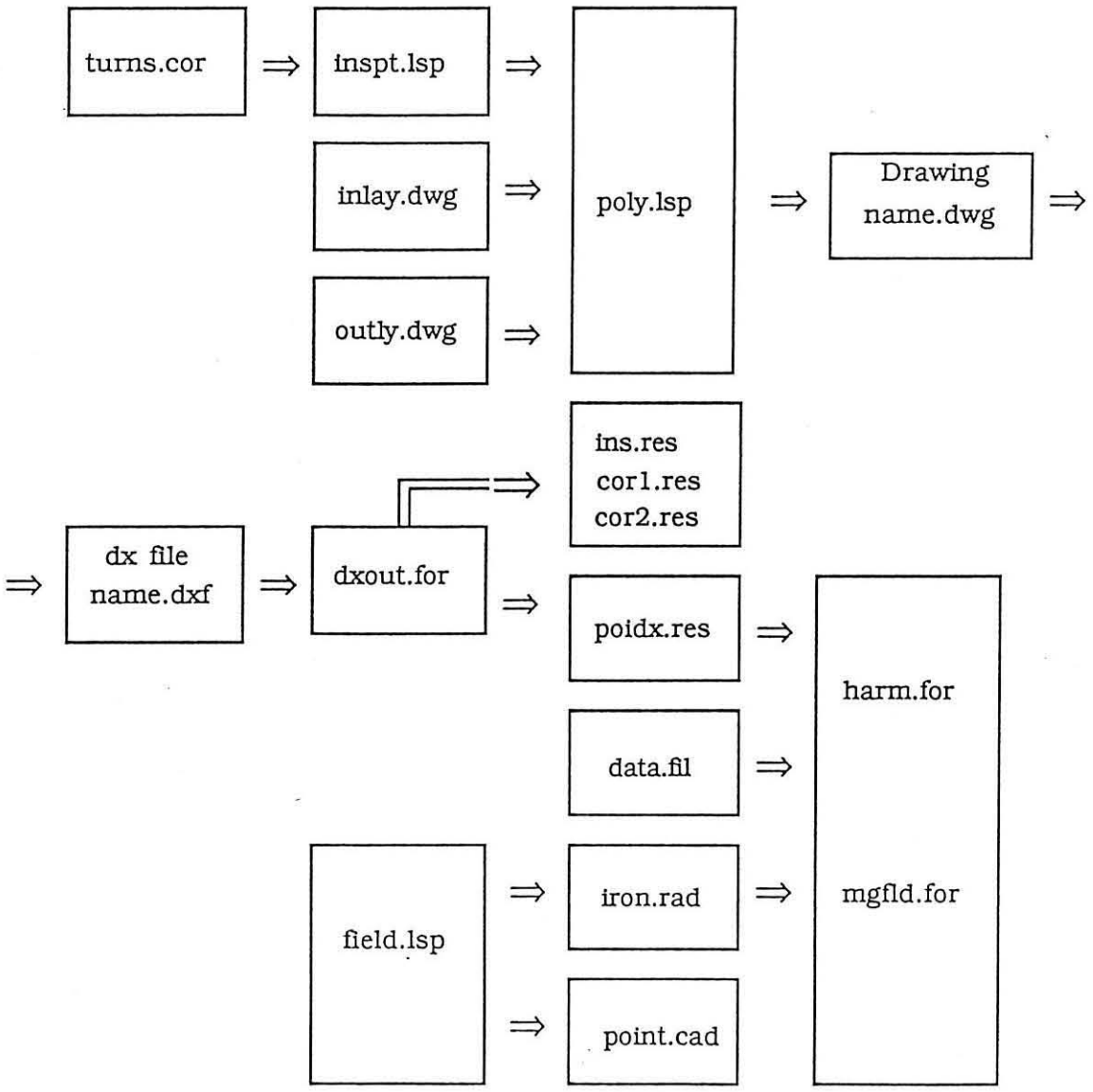
data.fil - Stores the permeability and the current values for a specific design.

turns.corr - Stores the turns corners coordinates.

poidx.res - Stores the new turns corners coordinates.

5. Data Flow

For a better (and easier) understanding of the whole system, we show the data flow, as has been built thus far:



6. Summary and Conclusions

As we have pointed out at the beginning of this report, this work represents only a start in the building of an entire drawing and analysis system. We now realize the power and capabilities of the system and have new ideas about a better use of the system. Without taking in to account further developments, which may require a lot of work (and time), there are some points which can be improved upon with very little effort.

Actually, the only data needed as input by the system is the file insrt.ang. A simple procedure can be developed, which should translate the corners coordinates (this is an output of the existing magnet analysis program PK) into the required angles.

The radius of the inner and outer layer should be supplied to the AutoCAD system through an external file, and this will make the poly.lsp file more general.

The conductor-type submenu has to be an icon menu type, to allow visual selection of the chosen conductor shape.

The special designed menu must be a submenu of the AutoCAD; there was no reason to do that now, but when this submenu is close to the final format, it will be necessary.

As we have already mentioned, the work with AutoCAD on SUN will be much easier, much faster, and more practical. For example, on the SUN, it will be easy to show, in a window, parameters of the magnet, such as the current density, the iron radius, the permeability, the magnetic field at any desired point, and so on. This will be accomplished by creating a file with these features and showing it in a window along side of the magnet drawing.

APPENDIX A

List of Magnet Cross Section Drawing With Computed Fields and Harmonics

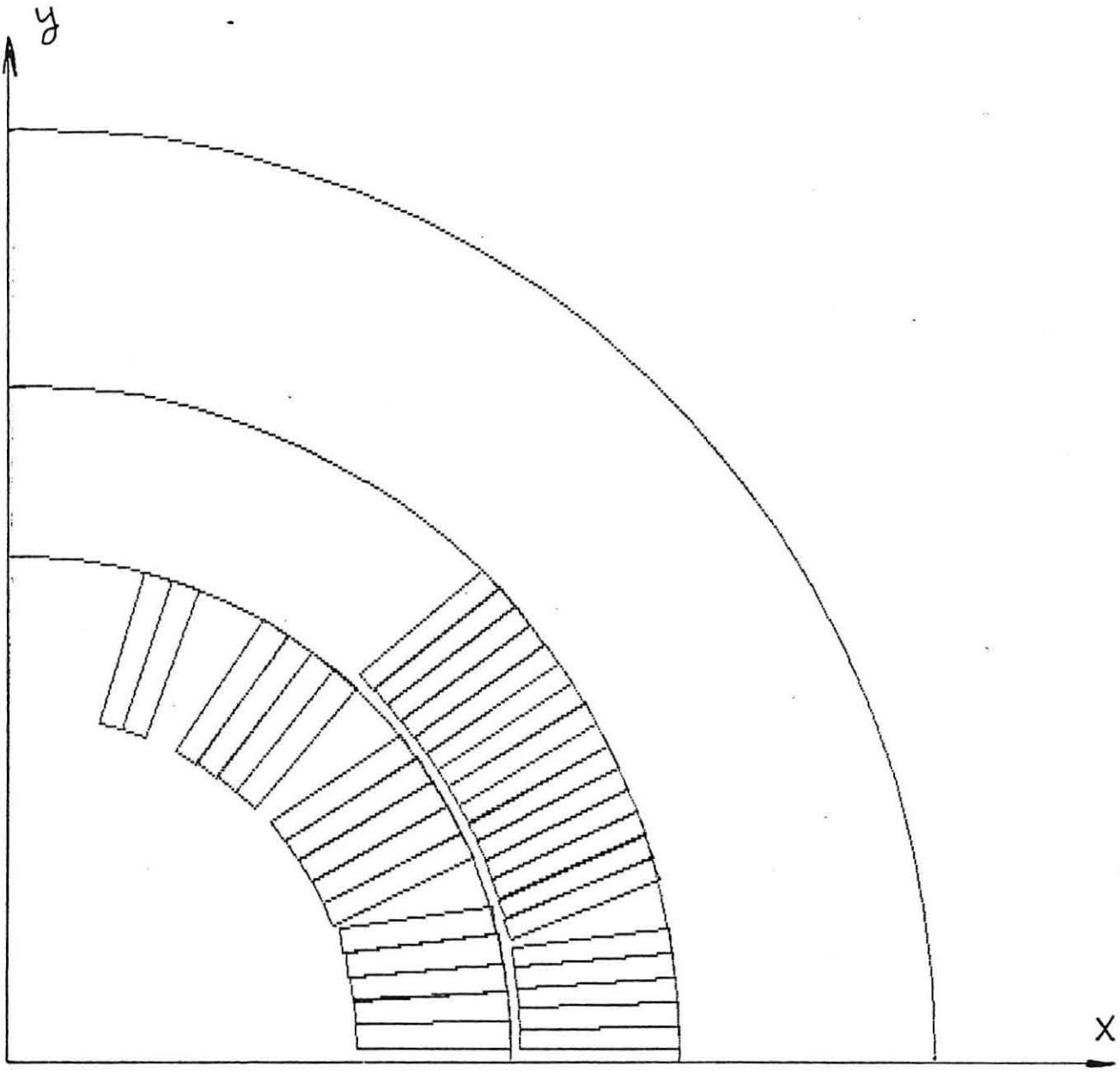


Fig. 1. Cross section of the NC-9 SSC dipole magnet.

```
R =  5.558 FOR IRON SHIELD

FIELD AT  X =  1.000  Y =  1.000

                      Bx                By                ABS-B
DIRECT(GAUSS)    3.61232658D+01   -5.07179841D+04   5.07179970D+04
 IMAGE(GAUSS)   -6.06418666D+01   -1.54123318D+04   1.54124511D+04
 TOTAL(GAUSS)   -2.45186007D+01   -6.61303159D+04   6.61303204D+04



NON-SKEW FIELD HARMONICS OF QUADRILATERAL BLOCK


DIPOLE

R =  5.558000 FOR IRON

TRANSFER FUNCTION =  -1.03341325D+01(G/A)

HARMONIC     DIRECT(GAUSS)     IMAGE(GAUSS)      TOTAL(GAUSS)          UNITS
    1      -5.07281687D+04  -1.54102795D+04  -6.61384482D+04   1.00000000D+04
    3       2.28678938D+01  -3.03079152D+01  -7.44002141D+00   1.12491623D+00
    5      -3.30200356D-01   5.12794738D-01   1.82594382D-01  -2.76079024D-02
    7       2.04778531D-02   3.25163888D-03   2.37294920D-02  -3.58785133D-03
    9       9.54138234D-02  -6.58406502D-05   9.53479828D-02  -1.44164227D-02

                                              *** END OF RUN ***
```

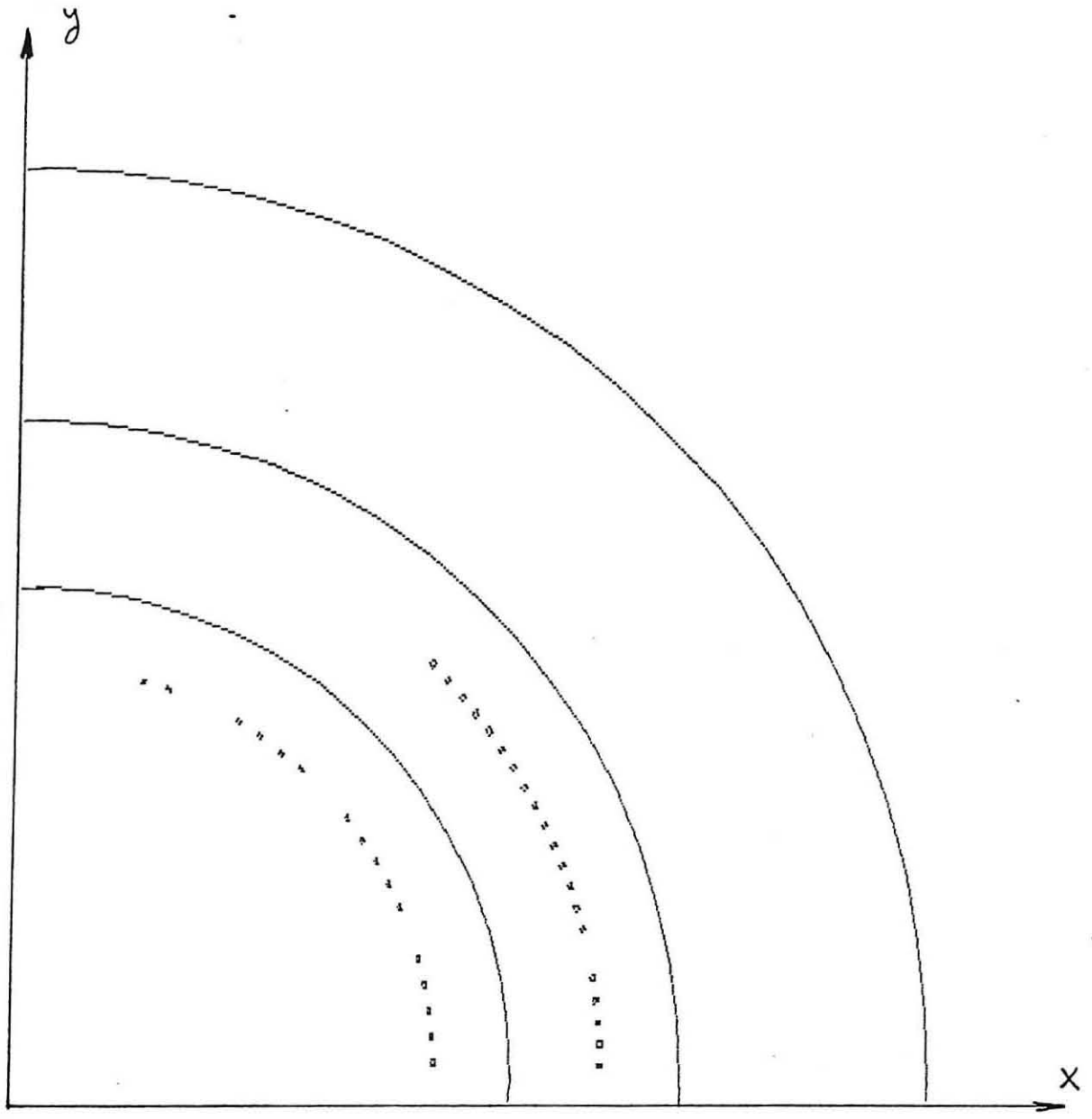


Fig. 2. Cross section for the NC-9 ; the conductor is replaced by a little box in the trapezoid center.

Main Field and Harmonics for Fig. 2 Configuration.

```
R =  5.558 FOR IRON SHIELD

FIELD AT  X =  1.000  Y =  1.000

                      Bx                  By                  ABS-B
DIRECT(GAUSS)   -2.93683094D+01    -5.05291545D+04     5.05291630D+04
 IMAGE(GAUSS)   -5.91262923D+01    -1.53511568D+04     1.53512707D+04
 TOTAL(GAUSS)   -8.84946017D+01    -6.58803113D+04     6.58803707D+04



NON-SKEW FIELD HARMONICS OF QUADRILATERAL BLOCK


DIPOLE

R =  5.558000 FOR IRON

TRANSFER FUNCTION =  -1.02941693D+01(G/A)

HARMONIC    DIRECT(GAUSS)     IMAGE(GAUSS)      TOTAL(GAUSS)          UNITS
    1     -5.05334041D+04  -1.53492791D+04  -6.58826832D+04   1.00000000D+04
    3     -1.57409258D+01  -2.95525293D+01  -4.52934551D+01   6.87486497D+00
    5     -1.05139106D+00   4.69228171D-01  -5.82162886D-01   8.83635665D-02
    7     -1.15998766D+00   2.65256650D-03  -1.15733509D+00   1.75666053D-01
    9     -2.26021715D-01  -5.00643035D-05  -2.26071779D-01   3.43142944D-02

                                          *** END OF RUN ***
```

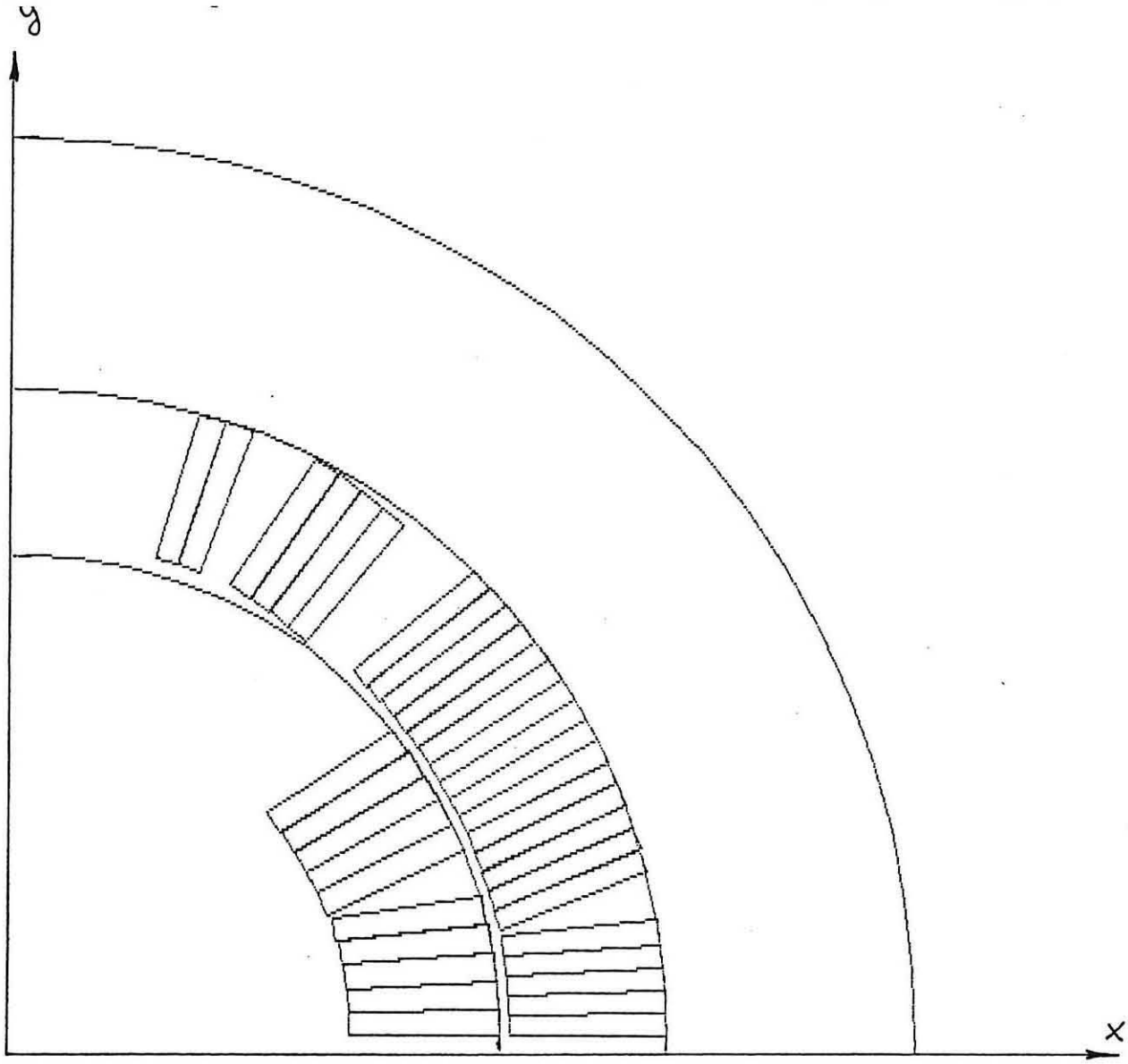


Fig. 3. Modified cross section for NC-9 configuration.

Main Field and Harmonics for Fig. 3 Configuration.

```
R =  5.558 FOR IRON SHIELD

FIELD AT  X =  1.000  Y =  1.000

                      Bx                  By                  ABS-B
DIRECT(GAUSS)   -2.45640371D+03    -4.89612508D+04     4.90228314D+04
 IMAGE(GAUSS)   -5.02320984D+00    -1.57426666D+04     1.57426674D+04
 TOTAL(GAUSS)   -2.46142692D+03    -6.47039174D+04     6.47507185D+04



NON-SKEW FIELD HARMONICS OF QUADRILATERAL BLOCK


DIPOLE

R =  5.558000 FOR IRON

TRANSFER FUNCTION =  -1.00485725D+01(G/A)

HARMONIC     DIRECT(GAUSS)      IMAGE(GAUSS)      TOTAL(GAUSS)              UNITS
    1      -4.85688692D+04   -1.57419945D+04   -6.43108637D+04    1.00000000D+04
    3      -1.14527642D+03   -2.50046595D+00   -1.14777688D+03    1.78473250D+02
    5       8.89368701D+01    1.67972201D-01    8.91048423D+01   -1.38553329D+01
    7       1.73471945D+01    2.77986544D-03    1.73499744D+01   -2.69782948D+00
    9      -2.28653775D+00   -1.07157719D-05   -2.28654847D+00    3.55546222D-01

                                                *** END OF RUN ***
```

APPENDIX B

List of Program Sources

Main Field and Harmonics for Fig. 3 Configuration.

```
R =  5.558 FOR IRON SHIELD

FIELD AT  X =  1.000  Y =  1.000

                      Bx                  By                 ABS-B
DIRECT(GAUSS)   -2.45640371D+03    -4.89612508D+04     4.90228314D+04
 IMAGE(GAUSS)   -5.02320984D+00    -1.57426666D+04     1.57426674D+04
 TOTAL(GAUSS)   -2.46142692D+03    -6.47039174D+04     6.47507185D+04
```

```
NON-SKEW FIELD HARMONICS OF QUADRILATERAL BLOCK


DIPOLE

R =  5.558000 FOR IRON

TRANSFER FUNCTION =  -1.00485725D+01(G/A)

HARMONIC    DIRECT(GAUSS)     IMAGE(GAUSS)      TOTAL(GAUSS)              UNITS
    1     -4.85688692D+04  -1.57419945D+04   -6.43108637D+04    1.00000000D+04
    3     -1.14527642D+03  -2.50046595D+00   -1.14777688D+03    1.78473250D+02
    5      8.89368701D+01   1.67972201D-01    8.91048423D+01   -1.38553329D+01
    7      1.73471945D+01   2.77986544D-03    1.73499744D+01   -2.69782948D+00
    9     -2.28653775D+00  -1.07157719D-05   -2.28654847D+00    3.55546222D-01

                                                *** END OF RUN ***
```

```
(defun C:DRAWP ()  ; Draw polygons
   (setq a (open "insrt.ang" "r"))
   (setq stringbag (read-line a))
   (setq pradin 2.9692)  ; inner layer radius
   (setq pradout 4.010580)  ; outer layer radius
   (setq prad pradin)
   (setq layercheck 0.0)
   (setq refpt (list 0.0 0.0))  ; set origin
   (command "color" "green")
   (while stringbag  ; while not EOF
      (setq rotang (atof (cdr4str stringbag)))  ; set angle of rot
      (cond
          ((< rotang layercheck) (setq prad pradout )
                                 (setq layercheck 0.0)
                                 (command "color" "yellow"))
          ( t  (setq layercheck rotang))
      )
      (setq  plpt (polar refpt (atof stringbag) prad)) ; insert pnt
      (cond
         ((= prad pradout )
                (command "insert" "outly" plpt 1 1 rotang))
         ( t
      (command "insert" "inlay" plpt 1 1 rotang))
      )
      (setq stringbag (read-line a))
   )
   (command "color" "white")
   (close a)
)
(defun fbl (stringbag)  ; find next blank in string
   (cond
        ((null stringbag)  nil )  ;  Dead string
        ( t  (fbl1 stringbag 1))  ; string needs work
        )
   )
(defun fbl1 (stringbag start )
   (setq intblank (chr 32 )) ; int repr of blank
   (cond
        ((null stringbag) nil)  ;End of string
        ((= intblank (substr stringbag 1  1)) start)  ;  Found it
        ((= (strlen stringbag ) 0 ) nil)  ; Check dead string
        ( t (fbl1 (substr stringbag 2) (1+  start))) ; else recurse
        ) ;substr. blank & check
)
(defun eatblank (stringbag)
   (setq repblank (chr 32 ))
   (cond
        ((null stringbag) nil)  ;  Dead string
        ((null (substr stringbag 1 1)) nil )  ;  Nothing string
        ((/= (substr stringbag 1 1) repblank) stringbag) ; Done
        ( t  (eatblank (substr stringbag 2))) ; eat rest of string
        )
 )
; CDR for string-type.Returns rest of string
(defun cdr4str (stringbag)
```

```
;chop off prefixed blanks
   (setq intstring (eatblank stringbag))
   (setq count (fbl intstring)) ; find next blank
   (cond
        ((null count) nil) ; Nothing there
        ((zerop count) nil) ; still nothing
        ((< count 0) nil)
        ( t  (eatblank (substr intstring count )))
   ) ; return a fresh string
)
 ;CAR for string-type. Returns unevaluated ATOM
( defun car4str (stringbag)
  ; chop off prefixed blanks
   (setq intstring (eatblank stringbag))
   (setq count ( fbl intstring)) ; find the next blank
   (cond
        ((null count) nil)  ; Nothing there
        ((< count 0) nil)
        ((zerop count) (substr intstring 1 ))  ;  whole string
        ( t  (substr intstring 1 (- count 1))) ; chop this string
   )  ; return a string
)
(defun arcbound (pradin  refpt)  ;  layer boundary
   (command "circle"
                  (setq p refpt)
                  (setq p pradin)
   )
)
```

```
C  This program calculates the magnetic field
C  B*,B*image & (B* + B*image ),at a given point Z
      Program MAGNETField
$NOTRUNCATE
      IMPLICIT REAL*8 (A-H,O-Z)
      dimension corner(8),Z(5),ZC(5),B(4),Bimage(4)
      complex*16 Zpnt,B,Bimage,calcb,calcbimage
      complex*16 Btotal,BimageTotal
      complex*16 Z,ZC,BT(3)
      CHARACTER*13 PNAME(3)
      COMMON / VALS / Z,ZC
C  File "POIDX.RES" contains the x-y coordinates of
C  the turns' corners,as they are computed by the
C  "POLYGON" program( file "DXOUT.FOR" ).
C  File "POINT.CAD" contains the x-y coordinates of
C  the points magnetic field should be computed. This
C  file should be created by the " DXFOUT " command of
C  AUTOCAD, after pointing at the requested point.
C  File " DATA.FIL " contains the values of
C  permeability and current,as they (should) appear on
C  the drawing, and are computed by " POLYGON " program.
      open (1,file='poidx.res')
      open (2,FILE='point.cad')
      open (3,FILE='data.fil')
      open (5,FILE=' Iron.Rad')
      read (3,*) PERMAB,CURRENT
      read (5,*) RADIUS
      PNAME(1) = 'DIRECT(GAUSS)'
      PNAME(2) = ' IMAGE(GAUSS)'
      PNAME(3) = ' TOTAL(GAUSS)'
      NVT = 4
      do 30 i=1,4
C  Initialize the complex vectors B , Bimage
      B(i)=dcmplx(0.0D0)
      Bimage(i)=dcmplx(0.0D0)
   30 continue
      read (2,*) x,y
      a = PERMAB*CURRENT
      Zpnt=dcmplx(x,y)
      ZCpnt=DCONJG(Zpnt)
      R2P = RADIUS**2
C   Read the coord of the corners
      do 20 i=1,36
      read (1, '(8(F10.6))') (corner(j),j=1,8)
      do 10 j=1,NVT
      Z(j)=dcmplx(corner(2*j-1),corner(2*j))
      ZC(j)=DCONJG(Z(j))
   10 continue
C  Compute the area of each turn
      area = CalcArea(Z,NVT)
C  This DO-LOOP assigns to Z the proper value,according
C  to the quadrant the magnetic field should be computed:
C  1st quadrant Z=Z
C  2nd quadrant Z= -Z*
C  3rd quadrant Z= -Z
```

```
C  4th quadrant Z= Z*
      do 40 nq=1,4
      if (nq.eq.1) then
      do 50 j=1,4
      z(j)=z(j)
      zc(j)=DCONJG(z(j))
   50 continue
      else
      do 60 j=1,4
      z(j)=(-1)**(nq+1)*DCONJG(z(j))
      zc(j)=DCONJG(z(j))
   60 continue
      endif
      z(NVT+1)=z(1)
      zc(NVT+1)=zc(1)
      B(nq)=B(nq)+a*CalcB(z,zc,zpnt,NVT)/area
      If (radius.GT.(10.0E-5))
     +Bimage(nq)=Bimage(nq)+a*CalcBimage(z,zc,zpnt,NVT,R2P)/area
   40 continue
   20 continue
   70 continue
C  The summation of the magnetic fields takes in account
C  the sign of integration & current in each quadrant :
C  1st quadr: integration(-) x current(+) = (-)
C  2nd quadr: integration(+) x current(-) = (-)
C  3rd quadr: integration(-) x current(-) = (+)
C  4th quadr: integration(+) x current(+) = (+)
      Btotal=-B(1)-B(2)+B(3)+B(4)
      BimageTotal=-Bimage(1)-Bimage(2)+Bimage(3)+Bimage(4)
C  Initilize BT for printing
      BT (1) = Btotal
      BT (2) = BimageTotal
      BT (3) = Btotal + BimageTotal
      if (radius.GT.(10.0E-5)) then
      write (*,1001) RADIUS
      else
      write (*,1006)
      endif
      write (*,1002) x,y
      write (*,1003)
      write (*,1004) (PNAME(i),DREAL( BT(i)),-DIMAG( BT(i)),
     +CDABS( BT (i)), i = 1,3)
      write (*,1005)
      close(1)
      close(2)
      close (3)
      close (5)
      stop
 1001 FORMAT (1H ,/,' R = ',F6.3,' FOR IRON SHIELD')
 1002 FORMAT (1H ,/,' FIELD AT  X = ',F6.3,'  Y = ',F6.3)
 1003 FORMAT (1H ,/,22X,'Bx',16X,'By',16X,'ABS-B')
 1004 FORMAT (1H ,A13,3X,1P,D15.8,4X,D15.8,4X,D15.8)
 1005 FORMAT (/,/,/)
 1006 FORMAT (1H ,' NO IRON')
      end
```

```
C  This function calculates the magnetic field B*
       complex*16 function CalcB (Z,ZC,zpnt,N)
      dimension Z(5),ZC(5)
      complex*16 Z,ZC
      complex*16 DeltaZ,DeltaZC
      complex*16 zpnt,sum
      sum=dcmplx(0.0D0)
      do 100 k=1,N
      DeltaZ=Z(k+1)-Z(k)
      DeltaZC=DCONJG(DeltaZ)
      sum =sum+((Z(k)-zpnt)*DeltaZC/DeltaZ-ZC(k))*
     +CDLOG((Z(k)-zpnt)/(Z(k+1)-zpnt))
  100 continue
      CalcB=sum
      return
      end
C  this function calc the magnetic field B*image
       complex*16 function CalcBimage (Z,ZC,zpnt,N,R2P)
      dimension Z(5),ZC(5)
      REAL*8 R2P
      complex*16 Z,ZC
      complex*16 DeltaZ,DeltaZC
      complex*16 zpnt,sum
      sum=dcmplx(0.0D0)
      do 200 k=1,N
      DeltaZ=Z(k+1)-Z(k)
      DeltaZC=DCONJG(DeltaZ)
      sum = sum+0.5*(Z(k)+Z(k+1))/zpnt*DeltaZC+
     +R2P/(zpnt**2)*((ZC(k)-R2P/zpnt)*DeltaZ/DeltaZC-Z(k))
     +*CDLOG((R2P-ZC(k)*zpnt)/(R2P-ZC(k+1)*zpnt))
  200 continue
      CalcBimage=sum
      return
      end
C  This function computes the area (abs value) of a polygon
C  with 4 complex corners
      REAL*8 Function CalcArea(Z,N)
      complex*16 Z(5),sum
      sum=dcmplx(0.0D0)
      Z(N+1) = Z(1)
      do 300 k=1,N
      sum = sum +(Z(k+1)+Z(k))*DCONJG(Z(k+1)-Z(k))
  300 continue
      CalcArea= (2.5D-1)*DIMAG(sum)
      return
      end
```

File "Magnet.Mnu"

```
***SCREEN
[ MAGNET ](setvar "CMDECHO" 0);(VMON);(load "FIELD");+
(load "POLY");(setvar "MENUECHO" 1);$S=MAIN_MENU

**MAIN_MENU
[ MAGNET ]$S=

[COND_TYP]$S=CONDUCTOR_TYPE

[ASSEMBLY]$S=MAGNET_GEOMETRY

[ FIELDS ]$S=MAGNET_CHARACT

[ SAVE ]save

[ BYE ]quit
**CONDUCTOR_TYPE 3
[ REC4-J ]sh del inlay.dwg;sh del outly.dwg;+
sh copy Rec4-JIn.dwg inlay.dwg;sh copy Rec4-JOut.dwg outly.dwg;+
graphscr;

[ REC4-I ]

[BOX-TYPE]sh del inlay.dwg;sh del outly.dwg;+
sh copy Box4-JIn.dwg inlay.dwg;sh copy Box4-JOut.dwg outly.dwg;+
graphscr;

[ LAST ]$S=
**MAGNET_GEOMETRY 3
[COS-TETA]DRAWP;GETR;

[ WINDOW ]

[ OTHER ]

[ LAST ]$S=
**MAGNET_CHARACT 3
[SAVE-GEO]dxfout poi 6;shell dxout;graphscr;

[calc-B]GETZ;\shell mgfld;

[HARMONIC]shell harm;

[NEW-RAD]zoom extent;IRONCHG;change;+
(setq point (list 0.0 IRONRAD));;;+
\dxfout poi 6;shell dxout;graphscr;

[ LAST ]$S=
```

```
(defun C:GETZ ()
(setq a (open "POINT.CAD" "w"))
(setq b (getpoint "Choose point for comp : "))
(setq c (reverse b))
(setq b1 (car b))
(setq b2 (car c))
(setq string (strcat (rtos b1 1 8) "  " (rtos b2 1 8)))
(write-line string a)
(close a)
)
(defun C:GETR ()
(setq radius (getreal "Enter Iron Shell Rad (default: no iron):"))
(cond
    ((not radius)  (setq radius 10.0E-6)
                   (setq string (strcat "10.0E+32")))
    ((not (null radius))  (setq string (rtos radius 1 3)))
    (t radius)
)
(setq a (open "IRON.RAD" "w"))
(write-line string a)
(close a)
(setq refpt (list 0.0 0.0))
(arcbound radius refpt)
(command "color" "red")
(arcbound pradin refpt)
(arcbound pradout refpt)
(command "color" "white")
)
(defun C:IRONCHG ()
(setq a (open "IRON.RAD" "r"))
(setq string (read-line a))
(setq IRONRAD (atof string))
(close a)
)
```

File "Harm.For"

```
      PROGRAM BLOCK
$NOTRUNCATE
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 EYE,Z,DZ,DZR
C     CHARACTER*9 L
      COMMON / VALS / Z(5),DZ(4),DZR(4)
      COMMON / PARS / NVT,NHP,NH
C Add to orig
      DIMENSION TotalDirect(5),TotalImage(5),Total(5)
      DIMENSION CORNER(8)
      open (1,FILE = 'POIDX.RES')
      open (2,FILE = 'DATA.FIL')
      open (3,FILE = 'IRON.RAD')
      read (2,*) PERMAB,CURRENT
      read (3,*) RFE
*
      NVT = 4
       PI = (2.0D0)*DACOS(0.0D0)
      ROD = PI/(1.8D2)
      EYE = (0.0D0,1.0D0)
**
C     CALL DATE(L)
C     WRITE(*,9001) L
      WRITE(*,9005)
C 10    WRITE(*,9010)
C     READ(*,*) NHP
C add to orig
      NHP=1
      NP = 2*NHP
      write (*,9040)
      if (RFE.GT.(10.0E-5)) then
      write (*,9043) RFE
      ELSE
      write (*,9042)
      ENDIF
*
C 20    RMAX = 0.0D0
C     DO 24 NV=1,NVT
C 22    WRITE(*,9020) NV
C     READ(*,*) R, DEG
C     IF (NHP*DEG .GT. 9.0D1) GO TO 22
C     RMAX = DMAX1(R,RMAX)
C     Z(NV) = R*CDEXP(EYE*ROD*DEG)
C24    CONTINUE
C add to orig
      do 99 m = 1,5
      TotalDirect(m) = 0.0D0
      TotalImage(m) = 0.0D0
      Total(m) = 0.0D0
99    continue
      do 100 NumBlock = 1,36
      read (1,*) ( corner(j),j = 1,8)
      do 101 NV = 1,NVT
      Z(NV) = cmplx(corner(2*NV-1),corner(2*NV))
101   continue
```

```
*
      Z(NVT+1) = Z(1)
      DO 26 NV=1,NVT
       DZ(NV) = Z(NV+1) - Z(NV)
      DZR(NV) = DCONJG(DZ(NV))/DZ(NV)
26    CONTINUE
C Add to orig
      area = CalcArea(Z,NVT)
*
C30    IMAGE = 0
C     WRITE(*,9025)
C     READ(*,*) IMAGE
C     IF (IMAGE .EQ. 0) GO TO 40
C     WRITE(*,9030)
C     READ(*,*) RFE
C     IF (RFE .LT. RMAX) GO TO 30
C40    NP = 2*NHP
C     WRITE(*,9040) NHP, NP
C     IF (IMAGE .EQ. 0) WRITE(*,9042)
C     IF (IMAGE .NE. 0) WRITE(*,9043) RFE
C     WRITE(*,9060)
      DO 70 M=1,5
      NH = NHP*(2*M - 1)
      IF (NH .EQ. 1) CALL DIR1(HARD)
      IF (NH .EQ. 2) CALL DIR2(HARD)
      IF (NH .GT. 2) CALL DIRN(HARD)
      HARI = 0.0D0
C     IF (IMAGE .EQ. 0) GO TO 60
      IF (RFE.GT.(10.0E-5))
     +CALL DIRI(HARI,RFE)
60    HART = HARD + HARI
C     WRITE(*,9070) NH, HARD, HARI, HART
C Add to orig
      TotalDirect(m) = TotalDirect(m) + (current/area)*HARD
      TotalImage(m) = TotalImage(m) + (current/area)*HARI
      Total(m) = Total(m) + (current/area)*HART
*
70    CONTINUE
C add to orig
100   continue
      write (*,9850) (Total(1)/Current)
      write (*,9060)
      do 102 m = 1,5
      NH = NHP*(2*M-1)
      B = Total(m)/Total(1)*1.0D+4
      write (*,9070) NH, TotalDirect(m),TotalImage(m),Total(m),b
102   continue
*
C80    WRITE(*,9800)
C     READ(*,*) JUMP
C     IF (JUMP .EQ. 1) GO TO 10
C     IF (JUMP .EQ. 2) GO TO 20
C     IF (JUMP .EQ. 3) GO TO 30
C     IF (JUMP .EQ. 9) GO TO 90
C     GO TO 80
```

```
90     WRITE(*,9900)
C Add to orig
       close(1)
       close(2)
       close (3)
       STOP
*
9000   FORMAT(1H )
9001   FORMAT(1H ,/,1H ,27X,A9,//)
9005   FORMAT(1H ,'NON-SKEW FIELD HARMONICS',
      $' OF QUADRILATERAL BLOCK',//)
9010   FORMAT(1H ,/,' TYPE N TO DESCRIBE A 2N POLE ASSEMBLY')
9020   FORMAT(1H ,/,' TYPE R & DEG FOR VERTEX',I2)
9025   FORMAT(1H ,/,' TYPE 0 (ZERO) IF NO IRON',
      $' -- OTHERWISE 1')
9030   FORMAT(1H ,/,' TYPE R OF IRON')
C9040   FORMAT(1H ,/,' N =',I2,'  FOR',I3,' POLE',/)
 9042   FORMAT(1H ,/,' NO IRON')
C9043   FORMAT(1H ,/,' R =',F5.2,' FOR IRON')
C9060   FORMAT(1H ,//,' HARMONIC',7X,'DIRECT',11X,'IMAGE',
C     $13X,'TOTAL',7X,'FOR G/J',/)
C9070   FORMAT(3H    ,I3,1PD20.8,D17.8,D18.8)
C Add to orig
9040    FORMAT(1H ,/,' DIPOLE ',/)
9043    FORMAT(1H ,/,' R =',F10.6,' FOR IRON')
9060    FORMAT(1H ,//,' HARMONIC',4X,'DIRECT(GAUSS)',5X,
      +'IMAGE(GAUSS)',6X,'TOTAL(GAUSS)',13X,'UNITS')
9070    FORMAT(3H    ,I3,1PD20.8,D17.8,D18.8,D18.8)
*
9800   FORMAT(1H ,//,' NEW 2N-POLE, VERTICES, IMAGE,',
      $' OR TERMINATE -- 1, 2, 3, OR 9',/)
9850   FORMAT(1H ,/,' TRANSFER FUNCTION =',1PD17.8,'(G/A)')
9900   FORMAT(1H ,/,1H ,51X,'*** END OF RUN ***',/)
       END
       SUBROUTINE DIR1(HARD)
       IMPLICIT REAL*8(A-H,O-Z)
       COMPLEX*16 Z,DZ,DZR
       COMMON / VALS / Z(5),DZ(4),DZR(4)
       COMMON / PARS / NVT,NHP,NH
       HARD = 0.0D0
       DO 10 NV=1,NVT
       HARD = HARD + DIMAG
      $(((Z(NV+1)*DCONJG(Z(NV))-Z(NV)*DCONJG(Z(NV+1)))/DZ(NV))
      $*CDLOG(Z(NV+1)/Z(NV)))
10     CONTINUE
       HARD = (4.0D-1)*HARD
       RETURN
       END
       SUBROUTINE DIR2(HARD)
       IMPLICIT REAL*8(A-H,O-Z)
       COMPLEX*16 Z,DZ,DZR
       COMMON / VALS / Z(5),DZ(4),DZR(4)
       COMMON / PARS / NVT,NHP,NH
       HARD = 0.0D0
       DO 10 NV=1,NVT
```

```
      HARD = HARD + DIMAG
     - $(DZR(NV)*CDLOG(Z(NV+1)/Z(NV)))
10    CONTINUE
      HARD = (8.0D-1)*HARD
      RETURN
      END
      SUBROUTINE DIRN(HARD)
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 Z,DZ,DZR
      COMMON / VALS / Z(5),DZ(4),DZR(4)
      COMMON / PARS / NVT,NHP,NH
       COF = ((4.0D-1)*NHP)/((NH-1.0D0)*(2.0D0-NH))
      HARD = 0.0D0
      DO 10 NV=1,NVT
      HARD = HARD + DIMAG
     $(DZR(NV)*(Z(NV+1)**(2-NH) - Z(NV)**(2-NH)))
10    CONTINUE
      HARD = COF*HARD
      RETURN
      END
      SUBROUTINE DIRI(HARI,RFE)
      IMPLICIT REAL*8(A-H,O-Z)
      COMPLEX*16 Z,DZ,DZR
      COMMON / VALS / Z(5),DZ(4),DZR(4)
      COMMON / PARS / NVT,NHP,NH
       COF = ((4.0D-1)*NHP)/((NH+1.0D0)*(NH+2.0D0)*(RFE**(NH-2)))
      HARI = 0.0D0
      DO 10 NV=1,NVT
      HARI = HARI + DIMAG
     $(DZR(NV)*((Z(NV)/RFE)**(NH+2) - (Z(NV+1)/RFE)**(NH+2)))
10    CONTINUE
      HARI = COF*HARI
      RETURN
      END
C Addto orig
      REAL*8 Function CalcArea(Z,N)
      COMPLEX*16 Z(5),SUM
      SUM = DCMPLX(0.0D0)
      DO 300 k = 1,N
      SUM = SUM + (Z(k+1) + Z(k))*DCONJG(Z(k+1)-Z(k))
300   continue
      CalcArea = (2.5D-1)*DIMAG(SUM)
      RETURN
      end
*
```

```
; This program computes the polar angle PTANG of the insertion
;  points, and the rotation angle ROTANG of each insert block.
; The angles are writen on file "INSRT.ANG" , the input file for
; POLY.LSP.
(defun C:ANGLIST ()
   (setq car4str nil)
; open file "TURNS.COR" which contents the coordinates of
; the corners, and read one line at a time
   (setq a  (open "turns.cor" "r"))
   (setq stringbag (read-line a))
(setq b (open "insrt.ang" "w"))
   (while (not (null stringbag)) ;do the next while not eof
; read x,y 4 all corners(4)
       (setq x1 (atof stringbag))
       (setq y1 (atof (cdr4str stringbag)))
       (setq stringbag (cdr4str (cdr4str stringbag)))
       (setq x2 (atof stringbag))
       (setq y2 (atof (cdr4str stringbag)))
       (setq stringbag (cdr4str (cdr4str stringbag)))
       (setq x3 (atof stringbag))
       (setq y3 (atof (cdr4str stringbag)))
       (setq stringbag (cdr4str (cdr4str stringbag)))
       (setq x4 (atof stringbag))
       (setq y4 (atof (cdr4str stringbag)))
       (setq xa (/ (+ x1 x2) 2))
       (setq xb (/ (+ x3 x4) 2))
       (setq ya (/ (+ y1 y2) 2))
       (setq yb (/ (+ y3 y4) 2))
       (setq ptang (angtos (atan yb xb) 3 6)) ;computes ptang in rad
; computes rotang in degrees
       (setq rotang (angtos (atan (- yb ya) (- xb xa)) 0 6))
       (setq ang4block (strcat ptang "   " rotang)) ; make a string
       (write-line ang4block b) ; write this string on "INSRT.ANG"
       (setq stringbag (read-line a)) ; next string
   )
   (close b)
   (close a)
)
; The next subr are explained in "POLY.LSP"
(defun fbl (stringbag)
   (cond
        ((null stringbag) nil )
        ( t  (fbl1 stringbag 1))
        )
   )
(defun fbl1 (stringbag start )
   (setq intblank (chr 32 ))
   (cond
        ((null stringbag) nil)
        ((= intblank (substr stringbag 1  1)) start)
        ((= (strlen stringbag ) 0 ) nil)
        ( t (fbl1 (substr stringbag 2) (1+  start)))
        )
)
(defun eatblank (stringbag)
```

```
   (setq repblank (chr 32 ))
   (cond
        ((null stringbag) nil)
        ((null (substr stringbag 1 1)) nil )
        ((/= (substr stringbag 1 1) repblank) stringbag)
        ( t  (eatblank (substr stringbag 2)))
        )
 )
(defun cdr4str (stringbag)
   (setq intstring (eatblank stringbag))
   (setq count (fbl intstring))
   (cond
        ((null count) nil)
        ((zerop count) nil)
        ((< count 0) nil)
        ( t  (eatblank (substr intstring count )))
   )
)
( defun car4str (stringbag)
   (setq intstring (eatblank stringbag))
   (setq count ( fbl intstring))
   (cond
         ((null count) nil)
         ((< count 0) nil)
         ((zerop count) (substr intstring 1 ))
         ( t  (substr intstring 1 (- count 1)))
   )
)
```

File "Dxout.For"

```
C  This program searches the file "POI.DXF" which was generated
C  by the AUTOCAD "DXFOUT" command,and generates 5 new files:
C  "INS.RES" - Temporary file storing the parameters of the
C              inserted entities
C  "COR1.RES" - Temporary file storing the coord & angle of the
C               ins points for the inner turns
C  "COR2.RES" - Temporary file storing the coord & angle of the
C               ins points for the outer turns
C  "POIDX.RES"- Output file which contains the coord of the corners
C               for all turns
C  "IRON.RAD" - Output file storing the iron radius
C  the program should generate (also) "DATA.FIL",containing the
C  attributes associated with the drawing( permeability
C  and current intensity).
       Program POLYGON
$NOTRUNCATE
      dimension delta (4,2)
      character findname *6
      character*(*) check1,check2
      integer unit,plabel,pend
      parameter (check1 = 'ENTITI', check2 = 'OUTLY')
      open (1,FILE='ins.res',Status='Old')
      OPEN (2,FILE='cor1.res',STATUS='Old')
      Open (3,FILE='cor2.res',Status='Old')
      Open (4,FILE = 'iron.rad',Status ='Old')
      Open (10,File='POI.DXF')
      Open (20,File='POIDX.RES',Status='Old')
  101 Do 100 Nline=1,100
C  Search the file "POI.DXF" (associated with the drawing)
C  for the begining of INSERT SECT
      Read (10,'(A6)') FindName
      If (FindName.Eq.'INSERT')  goto 102
  100 Continue
      GOTO 101
  102 plabel=0
  202 Do 200 Nline=1,100
      incheck=0
      if (findname.eq.'INSERT') incheck=1
      if (findname.eq.'VERTEX') incheck=1
      if (incheck.eq.1) then
  212 do 222 i=1,10
      plabel=0
C  write on temp file "INS.RES" coord & angle of ins pnt &
C  corners of ins blks (inner,outer)
      Call ParFinder (10,1,plabel,1000,FindName,check1)
      if (plabel.eq.1) goto 232
  222 continue
      GOTO 212
      endif
  232 if (findname.eq.'ENTITI') goto 103
      Read (10, '(A6)') FindName
  200 Continue
      GOTO 202
  103 unit=2
      plabel=0
```

```
  301 DO 300 Nline=1,100
C  writes on temp file "COR1.RES" & "COR2.RES" coord & angle
C  of ins pnt of inner,outer blks
      Call ParFinder (10,unit,plabel,1000,FindName,check2)
      if (plabel.eq.1) goto 104
  300 Continue
      GOTO 301
  104 if (unit.eq.3) goto 105
      plabel=0
      unit=3
      GOTO 301
  105 Continue
      close(10)
      Rewind 1
      Rewind 2
      Rewind 3
      plabel=1
C Compute the increments dx & dy 4 each corner of the inner block
      call dxcalc (1,delta,plabel)
      unit=2
      plabel=1
  106 do 400 Nline=1,16
      if (.not.eof(unit))
C Calculate the coord of the corners 4 the inner turns (first
C compute the rotated dx & dy 4 each inserted block)
     +call CornerCalc (unit,20,plabel,delta)
  400 continue
      if (.not.eof(unit)) goto 106
  107 plabel=1
C Compute the increments dx & dy 4 each corner of the outer block
      call dxcalc (1,delta,plabel)
  108 unit=3
      plabel=1
  109 do 500 nline=1,16
      if (.not.eof(unit))
C Calculate the coord of the corners 4 the outer turns (first
C compute the rotated dx & dy 4 each corner of inserted block
     +call CornerCalc (unit,20,plabel,delta)
  500 continue
      if (.not.eof(unit)) goto 109
      close(20)
       close(1)
       close(2)
       close(3)
      close (4)
 1000 stop
      end
C Compute dx & dy 4 coord system rotated by RotAng
      Function deltaxcalc (deltax,deltay,rotang)
      deltaxcalc = DeltaX*cos(RotAng) - DeltaY*sin(RotAng)
      Return
      end
      Function deltaycalc (deltax,deltay,rotang)
      deltaycalc = DeltaX*sin(RotAng) + DeltaY*cos(RotAng)
      Return
```

```
      end
C This subr searches 4 the block definition entities &
C the drawing entities
      Subroutine ParFinder (UnitR,UnitW,plabel,pend,FindName,CheckChar)
      integer UnitR,UnitW,plabel,pend
      character FindName *6,CheckChar *(*)
C Read a string; if 10 ,the next string is x-coord
C                if 20 ,the next string is y-coord
C                if 50 ,the next string is angle
      read (unitR,'(A6)',iostat=pend) FindName
      if (FindName.eq.' 10    ') then
         read (unitR,'(F10.6)',iostat=pend) point
         write (unitW,'(F10.6)') point
      endif
      if (FindName.eq.' 20    ') then
         read (unitR,'(F10.6)',iostat=pend) point
         write (unitW,'(F10.6)') point
      endif
      if (FindName.eq.' 50    ') then
         read (unitR,'(F10.6)',iostat=pend) point
         write (unitW,'(F10.6)') point
      endif
C If string=ENDBLK ,next is a new blk entiti or new section
      if (FindName.eq.'ENDBLK') plabel=1
C if string=ENTITI,write on "COR1.RES" 4 inner turns
C if string=OUTLY ,write on "COR2.RES" 4 outer turns
      if (FindName.eq.checkchar) plabel=1
      if (unitW.eq.3) then
C Do the loop again, 4 outer inserted blks
      plabel=0
      if (FindName.eq.'CIRCLE')  then
C If string = CIRCLE ,write on "IRON.RAD" the iron radius
      do 5 1 = 1,20
      read (unitR,' (A6)',iostat = pend) FindName
      if (FindName.eq.' 40    ') then
      read (unitR,' (F10.6)',iostat = plabel) point
      write (4,'(F10.6)') point
      plabel = 1
      GOTO 10
      endif
    5 continue
   10 continue
      endif
C You just finished the searching job
      endif
      return
      end
C This subr computes dx & dy 4 the corners of each inserted block,
C & rotates them by angle
      Subroutine DxCalc( unitR,delta,plabel)
      dimension InsPnt (3),delta(4,2)
      integer unitR,plabel
      real InsPnt
C Convert angle to Rad
      RadConv=174.532925E-4
```

```
      do 5 i=1,3
C Read x,y,angle of insertion pnt
      read (unitR,'(F10.6)',iostat=plabel) InsPnt(i)
    5 continue
      do 10 i=1,4
C Read x,y 4 each corner
      read (unitR,'(F10.6)',iostat=plabel) cornerX
      read (unitR,'(F10.6)',iostat=plabel) cornerY
C Compute dx,dy 4 each corner
      dx= InsPnt (1)-cornerX
      dy= InsPnt (2)-cornerY
C Rotate dx & dxy 4 all corners by RotAng[InsPnt(3)]
      delta(i,1) = deltaxcalc(dx,dy,InsPnt(3)*RadConv)
      delta(i,2) = deltaycalc(dx,dy,InsPnt(3)*RadConv)
   10 continue
      return
      end
C This subr computes the coord of the turns
      Subroutine CornerCalc (unitR,unitW,plabel,delta)
      dimension delta(4,2),InsPnt(3),corner(8)
      integer unitR,unitW,plabel
      real newx,newy,newdeltax,newdeltay,InsPnt
      RadConv=174.532925E-4
      do 5 i=1,3
C Read x,y & angle of inserted turns
      read (unitR,'(F10.6)',iostat=plabel) InsPnt(i)
    5 continue
      do 10 i=1,4
C Rotates dx & dy by angle
      newdeltax = deltaxcalc (delta(i,1),delta(i,2),InsPnt(3)*RadConv)
      newdeltay = deltaycalc (delta(i,1),delta(i,2),InsPnt(3)*RadConv)
      newx = InsPnt(1)-newdeltax
      newy = InsPnt(2)-newdeltay
C Computes x & y of all corners & fills the corner vector
      corner(2*i-1)=newx
      corner(2*i)=newy
   10 continue
      write (unitW,'(8(F10.6))') (corner(i),i=1,8)
      return
      end
```